APPENDIX F

Uucp Tutorial

CONTENTS

Uucp Tutorial

1. Introduction

The *uucp* network has provided a means of information exchange between UNIX[1] systems over the DDD network for several years [1]. This document is an attempt to provide the novice user with the background to make use of the network.

While it is perfectly reasonable for a user to assume that knowledge of the mechanism for transmission of information between UNIX Systems is unnecessary, in practice, it is often useful to understand basic principles so that the best possible use of the network can be made. Two earlier documents describe the goals and the details of an early implementation of *uucp [1, 2].* This document covers much of the same ground but reflects the improvements that have been made to the system and the information is tailored to the novice user.

The first half of the document discusses concepts. The second half explains the use of the user level interface to the network and provides numerous examples.

2. The Uucp Network

The *uucp(1)* network is a network of UNIX systems that allows file transfer and remote execution to occur on a network of UNIX Systems. The *extent* of the network is a function of both the interconnection hardware and the controlling network software. Membership in the network is tightly controlled via the software to preserve the integrity of all members of the network. The following sections describe the topology, services, operating rules, etc. of the network in detail to provide a framework for discussing use of the network.

2.1 Network Hardware

Uucp(1) was originally designed as a dialup network so that systems in the network could use the DDD network to communicate with each other. The three most common methods of connecting systems are,

1. Two UNIX Systems can be directly connected by cross-coupling (via a null modem) two of the computers ports. This means of connection is useful for only short distances (several hundred feet can be achieved although the RS232 standard specifies a much shorter distance) and is usually run at high speed (9600 baud). These connections run on asynchronous terminal ports.
2. A modem (using a private line or a limited distance modem) can be used to *directly* connect processors over a private line (using 103 or 212 type datasets).
3. Lastly, and more commonly, a processor can use a modem and an automatic calling unit (ACU) to use the DDD network to contact another system. This is by far the most common interconnection method and makes available the largest number of connections.

In principle *uucp* could be extended to use higher speed media (e.g., HYPERchannel[2], Ethernet[3], etc.) and this possibility is being explored for future UNIX releases. Some sites already support local modifications to *uucp* for using Datakit, X.25 (permanent virtual circuits) and calling through dataswitches.

---

1. UNIX is a Trademark of Bell Telephone Laboratories, Incorporated.
2. HYPERchannel is a Trademark of Network Systems Corporation.
3. Ethernet is a Trademark of Xerox Corp.

2.2 Network Topology

A large number of connections between systems are possible via the DDD network. The topology of the network is, however, determined by both the hardware connections and the software that controls the network. The next two sections deal with how that topology is controlled.

2.2.1 Hardware Topology As discussed earlier, it is possible to build a network using permanent or dial up connections. In Figure 1, a group of systems (A, B, C, D and E) are shown as connected via hardwired lines (all systems are assumed to have some answer only datasets so remote users or systems can be connected). A few systems have automatic calling units (K, D, F and G) and one system (H) does not have any capability for calling other systems. Users should be aware that the network consists of a series of point to point connections (A-B, B-C, D-B, E-B) even though it appears in Figure 1 that A and C are directly connected through B. The following observations can be made,

1. System H is isolated. It can be made part of the network by arranging for other systems to *poll* it at fixed intervals. This is an important concept to remember since transfers from systems that are *polled* will not leave the system until that system is called by a polling system.
2. Systems K, F, G and D can easily reach all other systems since they have calling units.
3. If system A (E or G) wishes to send a file to H (K, F or G) it must first send it to D (via system B) since D is the only system with a calling unit.

2.2.2 Software Topology The hardware capability of systems in the network defines the *maximum* number of connections in the network. The software at each node restricts the access by other systems and thereby defines the extent of the network. The systems of Figure 1 can be configured so that conceptually they appear as a network of systems that have equal access to each other or some restrictions can be applied. As part of the security mechanism used by *uucp(1)*, the extent of access that other systems have can be controlled at each node. Figure 2a,b shows how the network might appear at one node. Access is available from all systems in Figure 2a, however, in Figure 2b some of the systems have been configured to have greater or less access privileges than others (i.e., system C, E, G have one set of access privileges, systems F and B have another set, etc.).

Uucp uses the UNIX password mechanism coupled with a system file *(/usr/lib/uucp/L.sys)* and a filesystem permission file *(/usr/lib/uucp/USERFILE)* to control access between systems. The password file entries for *uucp* (usually, *luucp, nuucp, uucp, etc)* allow only those remote systems that know the passwords for these id's to access the local system. (Great care should be taken in revealing the password for these *uucp* logins since knowing the password allows a system to join the network.) The system file *(/usr/lib/uucp/L.sys)* defines the remote systems that a local host knows about. This file contains all information needed for a local host to contact a remote system (including system name, password, login sequence, etc.) and as such is protected from viewing by ordinary users.

In summary, while the available hardware on a network of systems determines the connectivity of the systems, the combination of *password* file entries and the *uucp* system files determines the extent of the network.

2.3 Forwarding

One of the recent additions to *uucp* (for UNIX System V) is a limited forwarding capability whereby systems that are part of the network can forward files through intermediate nodes. For example, in Figure 1 it is possible to send a file between node *A* and *C* through intermediate node *B*. For security reasons, when forwarding, files may only be transmitted to the *public* area (or fetched from the remote systems *public* area).

2.4 Security

The most critical feature of any network is the security that it provides. Users are familiar with the security that UNIX provides in protecting files from access by other users and in accessing the system via *passwords*. In building a network of processors, the notion of security is widened because access by a wider community of users is granted. Access is granted on a *system* basis (that is, access is granted to *all* users on a remote system). This follows from the fact that the process of sending (receiving) a file to (from) another system is done via daemons that use one special user id(s). This user id(s) is granted (denied) access to the system via the *uucp* system file *(/usr/lib/uucp/L.sys)* and the areas that the system has access to is controlled by another file *(/usr/lib/uucp/USERFILE)*. For example, access can be granted to the entire file system tree or limited to specific areas.

2.5 Software Structure

The *uucp(1)* network is a batch network. That is, when a request is made, it is spooled for later transmission by a daemon. This is important to users because the success or failure of a command will only be known at some later time via *mail* notification. For most transfers, there is little trouble in transmitting files between systems, however, transmissions are occasionally delayed or fail because a remote system can't be reached.

2.6 Rules of the Road

There are several rules by which the network runs. These rules are necessary to provide the smooth flow of data between systems and to prevent duplicate transmissions and lost jobs. The following sections outline what these rules are and what their influence on the network is.

2.6.1 Queuing Jobs submitted to the network are assigned a sequence number for transmission. Jobs are represented by a file (or files) in a common spool directory *(/usr/spool/uucp)*. When a file transfer daemon *(uucico)* is started to transmit a job, it selects a system to contact and then transmits *all* jobs to that system. Before breaking off the conversation, any jobs to be received from that remote system are accepted. The system selected as the one to contact is randomly selected if there is work for more than one system (in releases of *uucp(1)* prior to UNIX System V, the first system appearing in the spool directory was selected so preference was given to the most recently spawned jobs). *Uucp* may be sending to or receiving from many systems simultaneously. The number of incoming requests is only limited by the number of connections on the system and the number of outgoing transfers is limited by the number of ACUs (or direct connections).

2.6.2 Dialing and the DDD Network In order to transfer data between processors that are not directly connected, an auto dialer is used to contact the remote system. There are several factors that can make contacting a remote system difficult.

1. All lines to the remote system may be busy. There is a mechanism within *uucp* that restricts contact with a remote to certain times of the day (week) to minimize this problem.
2. The remote system may be down.
3. There may be difficulty in dialing the number (especially if a large sequence of numbers involving access through PBX's is involved). The dialing algorithm tries dialing a number twice and the algorithm used to dial remote systems is not perfect, particularly when intermediate dial tones are involved.

2.6.3 Scheduling and Polling When a job is submitted to the network an attempt to contact that system is made *immediately*. Only one conversation can exist between the same two systems at a time.

Systems that are *polled* can do nothing to force immediate transmission of data. Jobs will only be transmitted when the system is polled (hourly, daily, etc.) by a remote system.

2.6.4 Retransmissions and Hysterisis The *uucp* network is fairly tenacious in its attempt to contact remote systems to complete a transmission. To prevent *uucp* from continually calling systems that are unavailable, *hysterisis* is built into the algorithm used to contact other systems. This mechanism forces a minimum fixed delay (specifiable on a per system basis) to occur before another transmission can take place to that system.

2.6.5 Purging and Cleanup Users should be aware that transfers that cannot be completed after a defined period of time (72 hours is the value that is set when the system is distributed) are deleted and the user is notified.

2.7 Special Places: The Public Area

In order to allow the transfer of files to a system that a user may not have a login on, the *public* directory (usually kept in */usr/spool/uucppublic)* is available as an area with general access privileges. When receiving files in the *public* area, the user should dispose of them quickly as the administrative portion of *uucp* purges this area on a regular basis.

2.8 Permissions

2.8.1 File Level Protection In transferring files between systems, users should make sure that the destination area is writeable by *uucp*. The *uucp* daemons will preserve execute permission between systems and assign permission 0666 to transferred files.

2.8.2 System Level Protection The system administrator at each site determines the global access permissions for that processor. Thus, access between systems may be confined to only some sections of the filesystem by an administrator.

2.8.3 Forwarding Permissions The forwarding feature is a new addition to the *uucp* package. Users of this feature should understand that,

1. If forwarding is attempted through a node that is running an old version of *uucp(1)*, the transmission will not succeed.
2. Nodes that allow forwarding can restrict the forwarding feature in several ways.
    a. Allow forwarding for only certain users.
    b. Prevent forwarding to certain destination nodes (e.g., nodeblia).
    c. Allow forwarding for selected source nodes.
3. The most important restriction is that forwarding is allowed only for files sent to or fetched from the *public* area.

3. Network Usage

The following sections discuss the user interface to the network and give examples of command usage.

3.1 Name Space

In order to reference files on remote systems, a syntax is necessary to uniquely identify a file. The notation must also have several defaults to allow the reference to be compact. Some restrictions must also be placed on pathnames to prevent security violations. For example, pathnames may not include ".." as a component because it is difficult to determine whether the reference is to a restricted area.

3.1.1 Naming Conventions *Uucp* uses a special syntax to build references to files on remote systems. The basic syntax is,

system-name!path-name

where the *system-name* is a system that *uucp(1)* knows about. The *path-name* part of the name

may contain any of the following,

1. A fully qualified *path-name* such as

   xnode!/usr/youlfile

   The *path-name* may also be a directory name as in

   xnode!/usr/youldirectory

2. The login directory on a remote may be specified by use of the ~ character. The combination *~user* references the login directory of a user on the remote. For example,

   xnode!~admlfile

   would expand to

   xnode!/usr/sys/admlfile

   if the login directory for user *adm* on the remote system is */usr/sys/adm*.

3. The *public* area is referenced by a similar use of the prefix *~/user* preceding the pathname. For example,

   xnode!~/youlfile

   would expand to

   xnode!/usr/spool/uucp/youlfile

   if */usr/spool/uucp* is used as the spool directory.

4. Pathnames not using any of the combinations or prefixes discussed above are prefixed with the current directory (or the login directory on the remote). For example,

   xnode!file

   would expand to

   xnode!/usr/youlfile

The naming convention can be used in reference to either the *source* or *destination* file names.

3.2 Forwarding Syntax

The newest feature of *uucp* is the ability of *uucp* to allow files to be passed between systems via intermediate nodes. This is done via a variation of the *bang (!)* syntax that describes the path to be taken to reach that file. For example, a user on system *a* wishing to transmit a file to system *e* might specify the transfer as,

uucp file b!c!d!e!~/youlfile

if the user desires the request to be sent through b, c and d before reaching e. Note that the pathname is the path that the file would take to reach node *e*. Note also that the destination *must* be specified as the *public* area. Fetching a file from another system via intermediate nodes is done similarly. For example,

uucp b!c!d!e!~/youlfile x

fetches *file* from system *e* and renames it *x* on the local system. The forwarding prefix is the path *from* the local system not the path from the remote to the local system. The forwarding feature may also be used in conjunction with remote execution. For example,

uux xnode!uucp ynode!pnode!/usr/spool/uucppublic/file x

sends a request to *xnode* to execute the *uucp* command to copy a file from *pnode* to *x* on *xnode*.

3.3 Types of Transfers

Uucp(1) has a very flexible command syntax for file transmission. The following sections give examples of different combinations of transfers.

3.3.1 Transmissions of Files to a Remote Any number of files can be transferred to a remote system via *uucp(1)*. The syntax supports the *, ? and *[.] meta* characters. For example,

*uucp *.[ch] xnode!dir*

transfers *all* files whose name ends in *c* or *h* to the directory *dir* in the users login directory on *xnode*.

3.3.2 Fetching Files From a Remote Files can be fetched from a remote system in a similar manner. For example,

uucp xnode!.[ch] dir*

will fetch all files ending in *c* or *h* from the users login directory on *xnode* and place the copies in the subdirectory *dir* on the local system.

3.3.3 Switching Transmission of files can be arranged in such a way that the local system effectively acts as a switch. For example,

uucp ynode!files xnode!filed

will fetch *files* from the users login directory on *ynode*, rename it as *filed* and place it in the login directory on *xnode*.

3.3.4 Broadcasting Broadcast capability (that is, copying a file to many systems) is *not* supported by *uucp*, however, it can be simulated via a shell script as in

```
for i in xnode ynode mhtsd
do
        uucp file $i!broad
done
```

Unfortunately, one *uucp* command is spawned for each transmission so that it is not possible to track the transfer as a single unit.

3.4 Remote Executions

The remote execution facility allows commands to be executed remotely. For example,

uux "!diff xnode!/etc/passwd mhtsd!/etc/passwd > !pass.diff"

will *diff(1)* the password file on *xnode* and *mhtsd* and place the result in *pass.diff*.

3.5 Spooling

If a user wishes to continue modifying a file while a copy is being transmitted across the network, the *-c* option should be used. This forces a *copy* of the file to be queued. The default for *uucp* is not to queue copies of the files since it is wasteful of both cpu time and storage. For example, the following command will force the file *work* to be copied into the spool directory before it is transmitted.

uucp -c work xnode!~/you/work

3.6 Notification

The success or failure of a transmission is reported to users asynchronously via the *mail(1)* command. A new feature of *uucp* is to provide notification to the user in a file (of the users choice). The choices for notification are,

1. Notification returned to the requestors system (via the *-m* option). This is useful when the requesting user is distributing files to other machines and logging onto the remote machine to read mail is inconvenient.
2. A variation of the *-m* option is to force notification in a file (using the -mfile option where file is a file name). For example,

   uucp -mans /etc/passwd ynode!/dev/null

   will send the file */etc/passwd* to system *ynode* and place the file in the bit bucket *(/dev/null)*. The status of the transfer will be reported in the file *ans* as,

   uucp job 0306 (8/20-23:08:09) (0:31:23) /etc/passwd copy succeeded

3. *Uux(1)* always reports the exit status of the remote execution unless notification is suppressed (via the *-n* option). Notification can be sent to a different user on the remote (via the *-nuser* option).

3.7 Tracking and Status

The most pervasive change to the *uucp* package (for UNIX System V) has been revising the internal formatting of jobs so that each invokation of *uucp(1)* or *uux(1)* corresponds to a single job. It is now possible to associate a single job number with each command execution so that the job can be terminated or its status obtained.

3.7.1 The Job ID The default for the *uucp(1)* and *uux(1)* command is *not* to print the job number for each job. This was done for compatibility with previous versions of *uucp(1)* and to prevent the many shell scripts built around *uucp(1)* from printing job numbers. If the following environment variable

JOBNO=ON

is made part of the users environment and exported, *uucp(1)* and *uux(1)* will print the job number. Similarly, if the user wishes to turn job numbers off, the environment variable can be set as follows,

JOBNO=OFF

If the user wishes to force printing of job numbers without using the environment mechanism, the -j option can be used. For example,

```
uucp -j /etc/passwd ynode!/dev/null
uucp job 282
```

forces the job number (282) to be printed. If the -j option is not used, the ids of the jobs belonging to the user can be found by using the *uustat (1c)* command to obtain the job numbers. For example,

```
uustat
0282 tom ynode 08/20-21:47 08/20-21:47 JOB IS QUEUED
0272 tom ynode 08/20-21:46 08/20-21:46 JOB IS QUEUED
```

shows that the user has two jobs (282 and 272) queued.

3.8 Job Status

The *uustat (1)* command allows a user to check on one or all jobs that have been queued. The id printed when a job is queued is used as a key to query status of the particular job. An example of a request for the status of a given job is,

uustat -j0711

```
0711 tom ynode 07/30-02:18 07/30-02:18 JOB IS QUEUED
```

There are several status messages that may be printed for a given job; the most frequent ones are JOB IS QUEUED and JOB COMPLETED that have the obvious meanings. The manual page for *uustat (1)* lists the other status messages.

3.9 Network Status

The status of the last transfer to each system on the network can be found by using the *uustat (1c)* command. For example,

uustat -mall

will report the status of the last transfer to all of the
systems known to the local system.
The output might appear as,

```
znode   08/10-12:35   CONVERSATION SUCCEEDED
mnode   08/20-17:01   CONVERSATION SUCCEEDED
anode   07/22-16:31   DIAL FAILED
nodeb   08/20-18:36   WRONG TIME TO CALL
knode   08/20-20:37   LOGIN FAILED
```

where the status indicates the time and state of the last transfer to each system. When sending files to a system that has not been contacted recently, it is a good idea to use *uustat (1c)* to see *when* the last access occurred as the remote system may be down or out of service.

3.10 Job Control

With the unique job id generated for each *uucp (1)* or *uux (1)* command, it is possible to control jobs in the following ways.

3.10.1 Job Termination A job that may consist of transferring many files from several different systems can be terminated using the *-k* option of *uustat (1)*. If any part of the job has left the system then only the *remaining* parts of the job on the local system will be terminated.

3.10.2 Requeuing a Job The *uucp (1)* package clears out its working area of jobs on a regular basis (usually every 72 hours) to prevent the buildup of jobs that cannot be delivered. To force the date of a job to be changed to the current date, thereby lengthening the time that *uucp* will attempt to transmit the job, the *-r* option can be used. It should be noted that the *-r* option does not impart

immortality to a job. Rather, it only postpones deleting the job during housekeeping functions until the next cleanup.

3.10.3 Network Names Users may find the names of the systems on the network via the *uuname(1)* command. Only the ***names*** of the systems in the network are printed.

4. Utilities That Use Uucp

There are several utilities that rely on *uucp(1)* or *uux(1)* to transfer files to other systems. The following sections outline the more important of these functions to increase awareness of the extent of the use of the network.

4.1 Mail

The *mail(1)* command uses *uux(1)* to forward mail to other systems. For example, when a user types

mail xnode!tom

the *mail(1)* command invokes *uux(1)* to execute *rmail* on the remote system (*rmail* is a link to the *mail(1)* command). Forwarding mail through several systems (e.g., mail a!b!tom) does not use the *uucp(1)* forwarding feature but is simulated by the *mail(1)* command itself.

4.2 Netnews

The *netnews(1)* command that is locally supported on many systems uses *uux(1)* in much the same way that *mail(1)* does to broadcast ***network mail*** to systems subscribing to news categories.

4.3 Other Applications

The Office Automation System (OAS) uses *uux(1)* to transmit electronic mail between systems in a manner similar to the standard *mail(1)* command. Some sites have replaced utilities such as *lpr(1), opr(1), etc.* with shell scripts that invoke *uux(1)* or *uucp(1)*. Other sites use the *uucp* network as a backup for higher speed networks (e.g., PCL, NSC HYPERchannel, etc.).

5. Conclusion

This document has outlined the rules that run the network and attempted to illustrate the use and power of *uucp*.

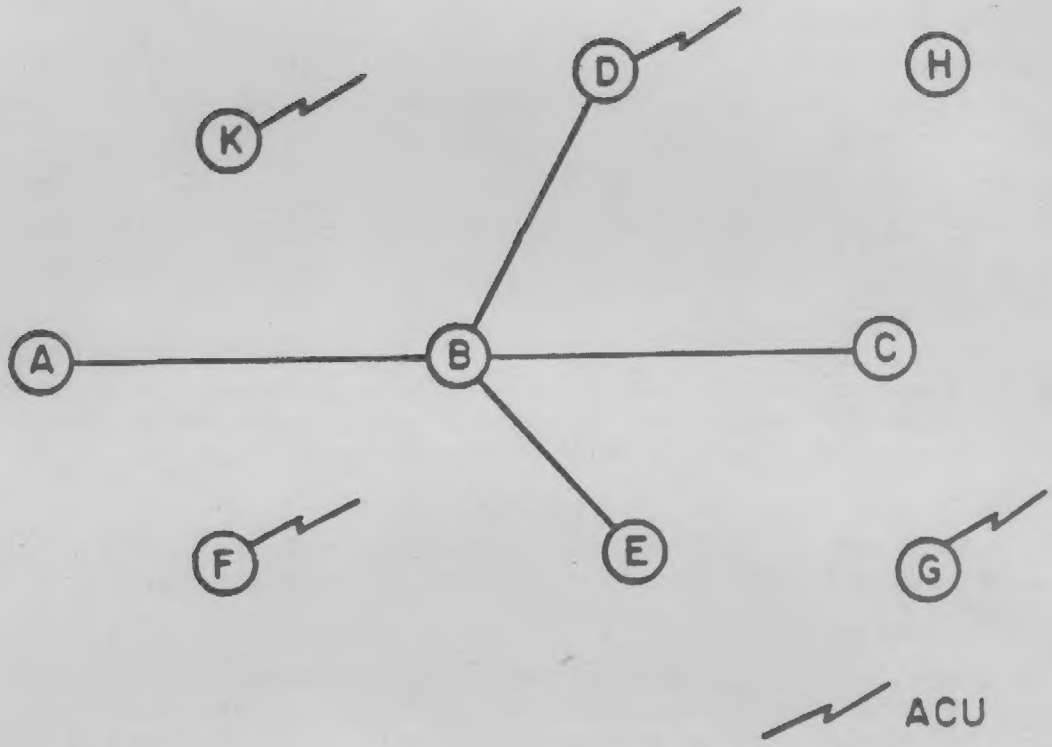

Figure 1 Uucp nodes.

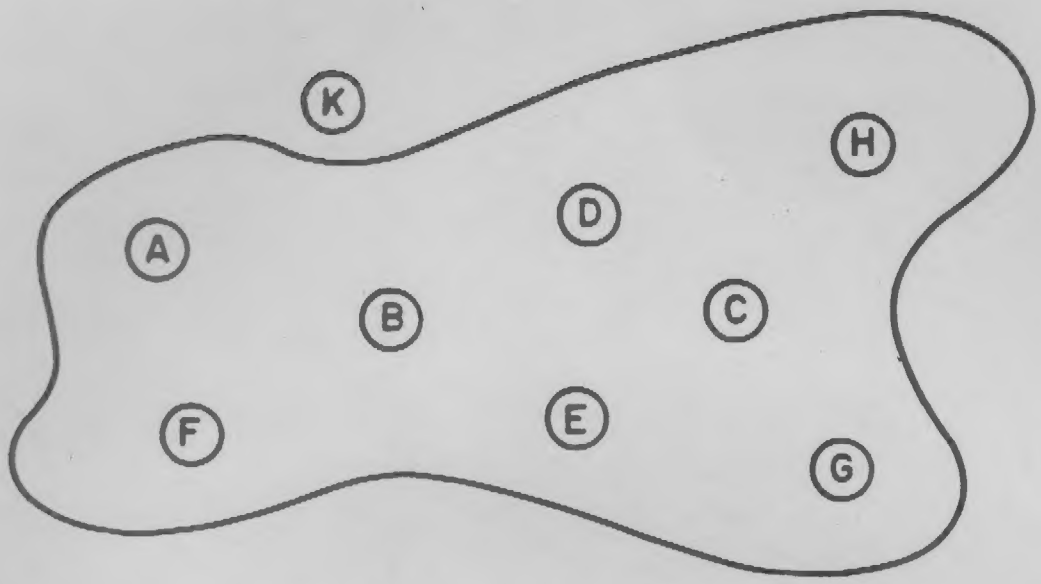

Figure 2a Uucp network excluding one node.

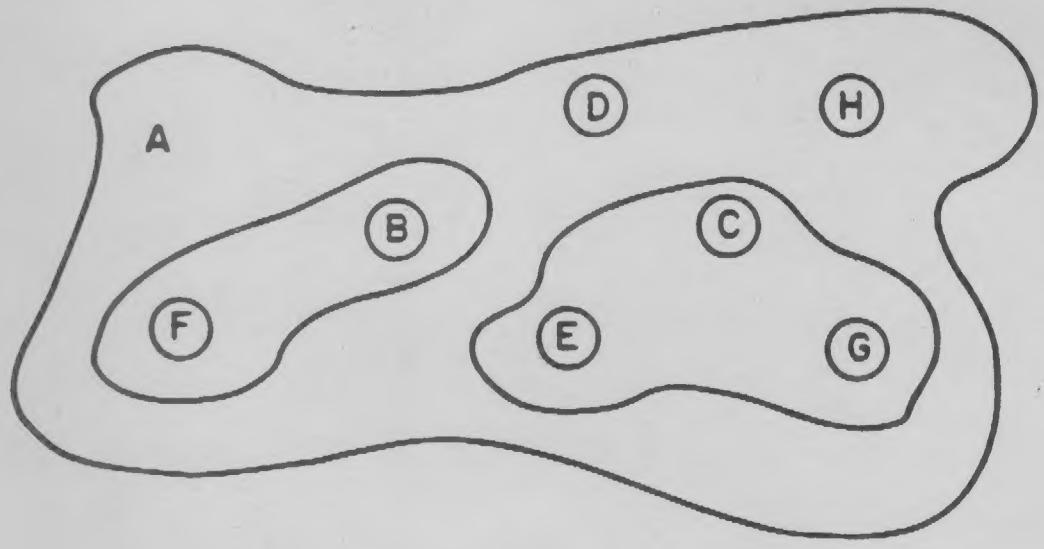

Figure 2b Uucp network with several levels of permissions.